ARTIFICIAL INTELLIGENCE

(CSC 462)

LAB # 12

NAME: MUAAZ BIN MUKHTAR

REG NO: FA21-BSE-045

CLASS & SECTION: BSSE-5A

SUBMITTED TO: SIR WAQAS ALI

DATE SUBMITTED: 04-12-2023

Department of Computer Science

Lab Task :

In a four queens problem, a board is a four-by-four grid of squares. A queen is a chess piece that can move on the chessboard any number of squares along any row, column, or diagonal. A queen is attacking another piece if, in a single move, it can move to the square the piece is on without jumping over any other piece. If the other piece is in the line of sight of the queen, then it's attacked by it). The four queens problem poses the question of how four queens can be placed on a chessboard without any queen attacking another queen. The problem is illustrated in figure:

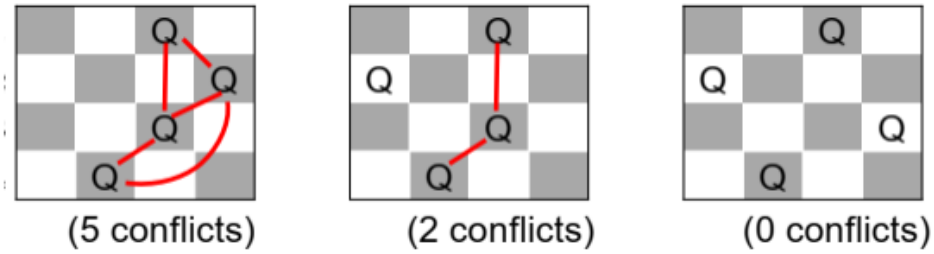

Figure 36 - 4 Queens problem

Code:

```python
def is_consistent(queen_positions, row, col):
    for i in range(row):
        if queen_positions[i] == col or \
           abs(i - row) == abs(queen_positions[i] - col):
            return False
    return True

def solve_four_queens(queen_positions, row):
    if row == 4:
        return [queen_positions[:]]  # Found a solution

    solutions = []

    for col in range(4):
        if is_consistent(queen_positions, row, col):
            queen_positions[row] = col
            solutions += solve_four_queens(queen_positions, row + 1)

    return solutions

def print_solution(solution):
    board = [["." for _ in range(4)] for _ in range(4)]
    for row, col in enumerate(solution):
        board[row][col] = "Q"

    for row in board:
        print(" ".join(row))

if __name__ == "__main__":
    queen_positions = [-1] * 4  # Initialize queen positions

    solutions = solve_four_queens(queen_positions, 0)

    if solutions:
        print_solution(solutions[0])
    else:
        print("No solution found.")
```

Output: